Amendments to the Claims:

This listing of claims will replace all prior version, and listings, of claims in the application.

Listing of Claims:

1 (Currently Amended) A method for providing data integration and exchange between a plurality of client applications over a network, wherein each of the client applications access a respective data source, and wherein the data sources of each of the client applications may be stored in different formats and not directly accessible by the other client applications, the method comprising~~ the steps of~~:

(a) providing an adapter API native to the client applications that provides a first set of methods for the client applications to use to translate data in the respective data sources into XML format; and

(b) modifying each of the client applications to invoke the first set of methods in the adapter API to convert the data in the respective data sources into XML format and to have the XML formatted data imported into a database on a server, thereby standardizing the data from the data sources.

2 (Currently Amended) The method of claim 1 further including step of:

(c) including a second set of methods in the adapter API for the client applications ~~to use~~ that provides consumption logic and methods for automatically exporting data defined in a Web-based schema registry, ~~export data~~ from the database into the client applications using ~~standard~~ Web services.

3 (Currently Amended) The method of claim 2 further including the step of: registering the respective data sources with a schema registry in order to create a schema definition and a document type definition (DTD).

4 (Canceled).

5 (Canceled).

6 (Currently Amended) The method of claim 3 wherein the adapter API includes an XML API comprising the first set of methods and a second set of methods, wherein the first set of methods comprises a Writer API, and the second set comprises a Reader API.

7 (Original) The method of claim 6 wherein the client applications are modified with generator logic that makes calls to methods comprising the adapter API, wherein once called, the Writer API converts data into XML format in memory and saves the XML format data in an XML file, which is then transported to the server.

8 (Canceled).

9 (Currently Amended) The method of claim ~~8~~ 7 wherein the adapter further includes verification code that verifies the generated XML data against the DTD defined in schema registry.

10 (Canceled).

11 (Original) The method of claim 7 wherein the server includes an import repository for receiving the XML files generated by the client applications.

12 (Original) The method of claim 11 wherein the server includes an XML loader that parses each of the XML files in the import repository and stores name/value pairs in the database according to a data hierarchy of the corresponding data source.

13 (Currently Amended) A computer-readable medium containing program instructions for providing data integration and exchange between a plurality of client applications over a network, wherein each of the client applications access a respective data source, and wherein the data sources of each of the client applications may be stored in different formats and not directly accessible by the other client applications, the program instructions for:

(a) providing an adapter API native to the client applications that provides a first set of methods for the client applications to use to translate data in the respective data sources into XML format; and

(b) modifying each of the client applications to invoke the first set of methods in the adapter API to convert the data in the respective data sources into XML format and to have the XML formatted data imported into a database on a server, thereby standardizing the data from the data sources.

14 (Currently Amended) The computer-readable medium of claim 13 further including instruction of:

(c) including a second set of methods in the adapter API for the client applications ~~to use~~ that provides consumption logic and methods for automatically exporting data defined in a Web-based schema registry, ~~export data~~ from the database into the client applications using ~~standard~~ Web services.

15 (Original) The computer-readable medium of claim 14 further including the instruction of: registering the respective data sources with a schema registry in order to create a schema definition and a document type definition (DTD).

16 (Canceled).

17 (Canceled).

18 (Currently Amended) The computer-readable medium of claim 15 wherein the adapter API includes an XML API comprising the first set of methods and a second set of methods, wherein the first set of methods comprises a Writer API, and the second set comprises a Reader API.

19 (Original) The computer-readable medium of claim 18 wherein the client applications are modified with generator logic that makes calls to methods comprising the adapter API, wherein once called, the Writer API converts data into XML format in memory and saves the XML format data in an XML file, which is transported to the server.

20 (Canceled).

21 (Currently Amended) The computer-readable medium of claim ~~20~~ 19 wherein the adapter further includes verification code that verifies the generated XML data against the DTD defined in schema registry.

22 (Canceled).

23 (Original) The computer-readable medium of claim 19 wherein the server includes an import repository for receiving the XML files generated by the client applications.

24 (Original) The computer-readable medium of claim 23 wherein the server includes an XML loader that parses each of the XML files in the import repository and stores name/value pairs in the database according to a data hierarchy of the corresponding data source.

25 (Currently Amended) A data integration system comprising:

a network;

a server coupled to the network, the server including a schema registry, a database, and a published adapter API native to the client applications that provides a first set of methods for translating data in respective data sources into XML format; and

a plurality of client applications coupled to the network and in communication with the server, wherein each of the client applications access ~~a~~ the respective data sources, and wherein the data sources of each of the client applications may be stored in different formats and are not directly accessible by the other client applications, and

wherein at least a portion of the client applications includes a corresponding schema definition and document type ~~document~~ definition (DTD) registered with the schema registry, and the portion of the client applications includes generation logic for making calls to the first set of methods in the adapter API, such that data in the respective data sources are converted into XML format and transferred to the server, wherein the XML data is verified against the corresponding DTD prior to storage in the database, thereby standardizing the data from the data sources.

26 (Currently Amended) The system of claim 25 wherein the adapter API further includes a second set of methods for the client applications ~~to invoke~~ that that provides consumption logic and methods for automatically exporting data defined in the schema registry, ~~export data~~ from the database into the client applications using ~~standard~~ Web services.

27 (Original) The system of claim 26 wherein the adapter API includes an XML API comprising the first and second methods, wherein the first set of methods comprises a Writer API and the second set comprises a Reader API.

28 (Original) The system of claim 27 wherein the server further includes a schema generator for generating the schema definition, a DTD generator for generating the DTD, and an adapter software kit that is downloaded from the server and used to incorporate the adapter API into the client applications.

29 (Original) The system of claim 28 wherein the server further includes an import repository for receiving XML files generated by the client applications.

30 (Original) The system of claim 29 wherein the server includes an XML loader that parses each of the XML files in the import repository and stores name/value pairs in the database according to a data hierarchy of the corresponding data source.